

ANÁLISIS EXPLORATORIO DE DATOS: ATENCIONES DE ASEGURADOS SIS EN EL PRIMER NIVEL DE ATENCIÓN EN LA RED DE SALUD AREQUIPA CAYLLOMA

Vladimir Barrios, Elker Garcia ,Vidal Alvarez

I. ESTRUCTURA DE LOS DATOS

DATA SIS

Data estructurada, con formato específico tabular (filas y columnas), que consta de 27 dimensiones:

- **NRO FORMATO:** Número de Formato Único de Atención (FUA). Documento que se genera para la atención del afiliado SIS.
- **F. ATENCION:** Fecha de atención. Indica el día en que se realizó la atención médica al beneficiario.
- **TIP. DOC.:** Tipo de documento. Especifica el tipo de documento de identidad del beneficiario, como DNI, carnet de identidad.
- **DOCUMENTO:** Número de documento de identidad del beneficiario. Este es el identificador único del paciente dentro del sistema.
- **CONTRATO:** Número o código del contrato de afiliación al Seguro Integral de Salud (SIS).
- **BENEFICIARIO:** Nombre del beneficiario, es decir, la persona que recibe la atención médica.
- **F. NACIMIENTO:** Fecha de nacimiento del beneficiario. Este dato es importante para calcular la edad del paciente y evaluar el contexto de la atención.
- **EDAD:** Edad del beneficiario al momento de la atención. Este campo se calcula a partir de la fecha de nacimiento y la fecha de atención.
- **SEXO:** Sexo del beneficiario, representado como 'M' para masculino o 'F' para femenino.

- EESS CODIGO: Código del Establecimiento de Salud (EESS). Este código identifica el lugar donde se brindó la atención.
- EESS NOMBRE: Nombre del Establecimiento de Salud del primer nivel de atención donde se realizó la atención. Es el nombre del centro de salud o puesto de salud.
- SERVICIO: Código o descripción del servicio médico proporcionado, como consulta externa, hospitalización, emergencia, etc.
- DNI PROFESIONAL: Número de DNI del profesional de salud que atendió al paciente. Es el identificador único del médico o personal de salud.
- NOMBRE PROFESIONAL: Nombre del profesional de salud que brindó la atención.
- TIPO PROFESIONAL: Tipo de profesional de salud (médico, enfermero, obstetra, etc.).
- TARIFA: Tarifa o costo asociado al servicio brindado, que puede variar según el tipo de atención o contrato. Dato depreciable porque la información que se genera no es real.
- HIST. CLINICA: Número de la historia clínica del paciente. Es un identificador interno del paciente en el sistema del Establecimiento de Salud.
- COMPONENTE: Componente del servicio o atención brindada.
- COND. MATERNA (*): Condición materna. Este campo se refiere a la condición de la madre en el caso de atenciones perinatales, donde se podría registrar si hay algún riesgo o complicación durante el embarazo o parto.
- TIP. ATENCION (**): Tipo de atención recibida, como ambulatoria, hospitalización, urgencias, entre otros. Define la naturaleza del servicio brindado.
- LUG. ATENCION (*):** Lugar de atención, que puede ser dentro del mismo Establecimiento de Salud o en otro lugar, como a domicilio o en otra entidad.
- EESS REFERENCIA: Establecimiento de salud de referencia. Indica si el paciente fue derivado a otro centro de salud para continuar con la atención o recibir un tratamiento específico.

- F. REGISTRO: Fecha de registro de la atención en el sistema. Esta fecha puede diferir de la fecha de atención y se refiere a cuándo se ingresó la información en el sistema.
- DIGITADOR: Nombre o código del digitador que ingresó la información al sistema.
- NRO CRED: Número de controles de crecimiento y desarrollo.
- AÑO: Año correspondiente al periodo de digitación.
- MES: Mes correspondiente al periodo de digitación.

DATA HIS (Data Semi-estructurada)

Data semi-estructurada, que a pesar de tener una estructura tabular, los códigos y diagnósticos pueden aparecer en distintas columnas o no pueden estar presentes (falta de uniformidad), la cual consta de 109 dimensiones:

- fecha atención: Fecha en que el paciente recibió la atención médica.
- lote: Número de lote o identificador que agrupa un conjunto de atenciones o registros. Puede estar relacionado con la entrada de datos en un proceso de carga masiva.
- pag: Número de página dentro de un documento físico o digital donde se registró la atención, si corresponde.
- reg: Número de registro.
- dni paciente: Número de Documento Nacional de Identidad (DNI) del paciente, utilizado para identificarlo de manera única.
- historia: Número de historia clínica del paciente, que es un identificador único dentro del sistema del Establecimiento de Salud.
- Edad(A): Edad del paciente en años al momento de la atención.
- Edad(M): Edad del paciente en meses. Este campo es especialmente relevante para pacientes pediátricos.

- Edad(D): Edad del paciente en días. Útil para recién nacidos y lactantes.
- finan: Tipo de financiamiento de la atención. Dato depreciable porque la información se ingresa de forma manual sin validación.
- Cest: Código del establecimiento de salud donde se brindó la atención.
- Cser: Código del servicio médico proporcionado.
- ups: Código de la Unidad Productora de Servicios (UPS), que indica la unidad específica dentro del establecimiento que brindó la atención (por ejemplo, pediatría, ginecología, etc.).
- descripción ups: Descripción de la Unidad Productora de Servicios. Es una versión más legible del código UPS.
- turno: Turno en el que se brindó la atención, como mañana, tarde, o noche.
- dni personal: Número de DNI del profesional de salud que atendió al paciente.
- nombre personal: Nombre completo del profesional de salud que brindó la atención.
- profesión: Profesión del personal de salud, como médico, enfermero, obstetra, etc.
- condición: Condición del paciente al momento de la atención, podría referirse a si estaba estable, crítico, en observación, etc.
- nombre registrador: Nombre de la persona que registró la atención en el sistema.
- ipress: Código del IPRESS (Institución Prestadora de Servicios de Salud), que identifica el establecimiento donde se brindó la atención.
- nombre ipress: Nombre del IPRESS o establecimiento de salud.
- microred: Código de la microred de salud a la que pertenece el establecimiento de salud.

- nombre microred: Nombre de la microred a la que pertenece el establecimiento de salud.
- items: Número total de ítems registrados en la atención.
- codigo1 a codigo28: Códigos de los diagnósticos asociados a la atención, generalmente siguiendo la codificación CIE-10 (Clasificación Internacional de Enfermedades).
- diag1 a diag28: Tipo del diagnóstico correspondiente a cada código CIE-10 (Definitivo, Recuperativo, Presuntivo)
- lab1 a lab28: Identificador relacionados a exámenes de laboratorio asociados con cada diagnóstico, si corresponde.

II. SELECCIÓN DE CARACTERÍSTICAS

DATA SIS

Las variables que has identificado en el Data SIS como importantes son clave para realizar un análisis preciso y eficiente. Estas variables te permitirán analizar aspectos críticos de las atenciones de salud financiadas por el SIS:

- **F. ATENCION (Fecha de Atención):**

Importancia: Permite analizar la temporalidad de las atenciones, identificar patrones estacionales, y realizar análisis de tendencias a lo largo del tiempo.

- **DOCUMENTO (Número de Documento):**

Importancia: Es el identificador único del paciente, esencial para realizar un seguimiento individualizado de las atenciones y analizar la frecuencia de visitas.

- **EDAD:**

Importancia: La edad es una variable fundamental para el análisis demográfico, que permite segmentar la población atendida y analizar riesgos específicos asociados a diferentes grupos etarios.

- **SEXO:**

Importancia: Es crucial para identificar diferencias en el acceso y la calidad de las atenciones entre hombres y mujeres, así como para realizar estudios epidemiológicos que diferencian entre géneros.

- DNI PROFESIONAL:

Importancia: Identifica al profesional de salud que brindó la atención, permitiendo analizar la distribución del trabajo entre los profesionales, así como su desempeño y eficiencia.

- EESS CODIGO (Código del Establecimiento de Salud):

Importancia: Permite identificar dónde se realizaron las atenciones, facilitando el análisis por ubicación geográfica, eficiencia de los establecimientos, y la carga de trabajo de cada centro.

- SERVICIO:

Importancia: Identifica el tipo de servicio médico proporcionado (como consulta externa, hospitalización, etc.), lo que es clave para entender la distribución de los recursos y la demanda de diferentes tipos de servicios.

- TIPO PROFESIONAL:

Importancia: Define el rol del profesional de salud (como médico, enfermero, obstetra), lo que es útil para analizar la distribución de atenciones por especialidad y la relación entre el tipo de profesional y los resultados de salud.

DATA HIS

En Data HIS, las variables que has identificado son cruciales para realizar un análisis detallado y comprender mejor las atenciones de salud en los establecimientos:

- fecha atención:

Importancia: Permite analizar la temporalidad de las atenciones, similar al análisis en Data SIS, ayudando a identificar patrones temporales y tendencias en la prestación de servicios de salud.

- dni paciente:

Importancia: Actúa como identificador único del paciente en Data HIS, permitiendo el seguimiento individual de cada paciente y la posibilidad de cruzar esta información con otros datasets.

- dni personal:

Importancia: Identifica al profesional de salud que atendió al paciente, facilitando el análisis de la distribución del trabajo y del desempeño de los profesionales en diferentes servicios y ubicaciones.

- ipress:

Importancia: Código de la Institución Prestadora de Servicios de Salud donde se brindó la atención. Es fundamental para identificar la ubicación geográfica y analizar la eficiencia y carga de trabajo de cada IPRESS.

- codigo1 a codigo28:

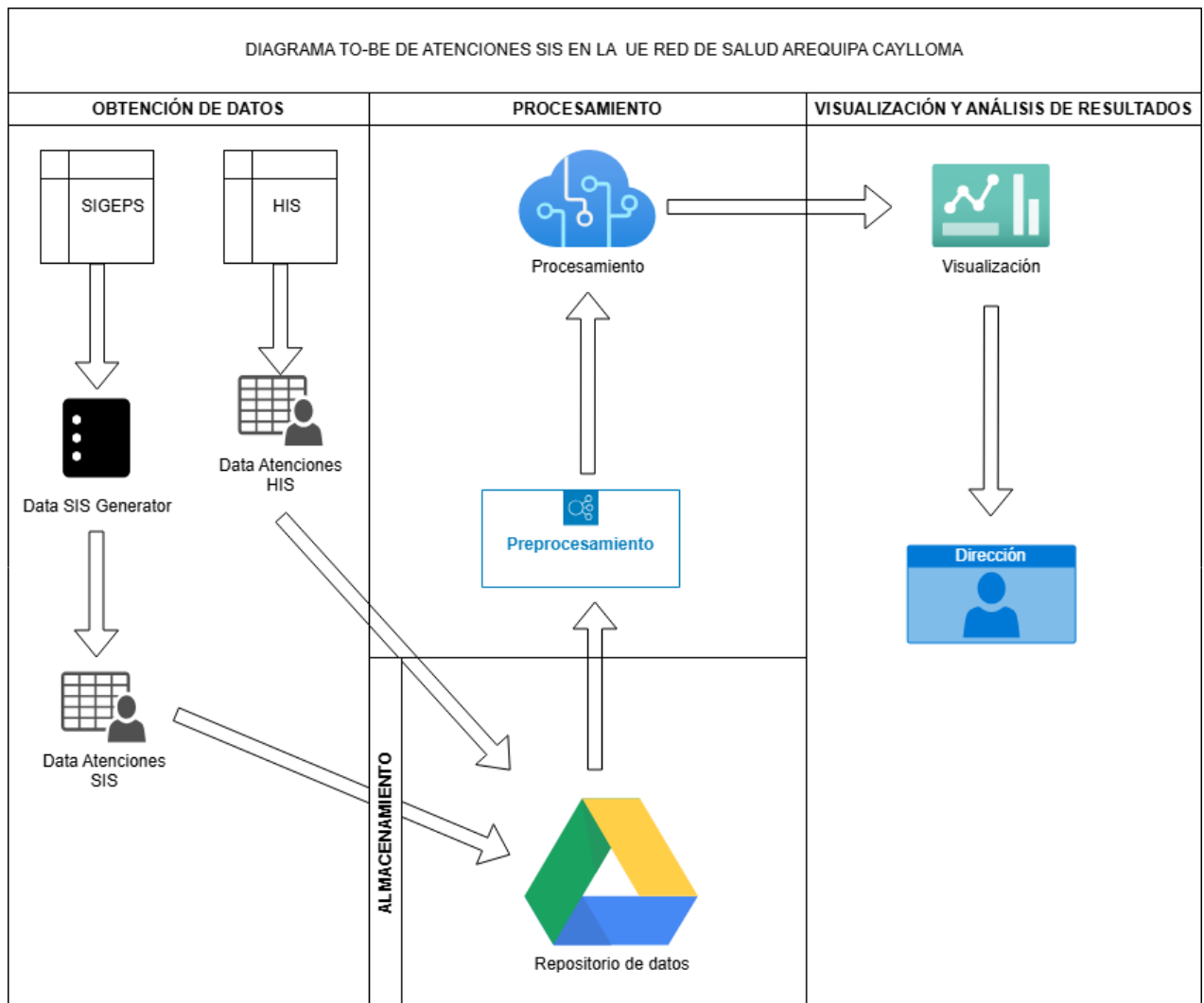
Importancia: Estos códigos representan los diagnósticos asociados a cada atención, generalmente utilizando la codificación CIE-10 (Clasificación Internacional de Enfermedades). Analizar estos códigos permite entender las condiciones médicas más prevalentes, realizar estudios epidemiológicos, y evaluar la complejidad de las atenciones.

III. EXPLORACIÓN DE PATRONES

| DATA SIS | DATA HIS |
|--------------------|----------------|
| F. ATENCION | fecha atención |
| DOCUMENTO | dni paciente |
| EESS CODIGO* | lpress |
| NOMBRE PROFESIONAL | dni personal |

- Cada dataset maneja códigos diferentes, pero es posible relacionar a través de un diccionario que identifique sus equivalencias con información que se maneja a nivel de la UE (RSAC).

IV. PROPUESTA DE IMPLEMENTACIÓN DE ANÁLISIS EXPLORATORIO DE DATOS



La propuesta estará enmarcada desde un enfoque de mejora continua; es decir, el proceso será cíclico, luego de culminado el análisis de resultados para la toma de decisiones de nivel directoral, se volverá a iniciar para identificar nuevos datos y por consiguiente generar nuevo conocimiento.

1. OBTENCIÓN DE DATOS

La Data HIS se genera directamente del sistema HIS-MINSA, a través de reportes mensuales generados en formato EXCEL, a nivel de UE (RSAC).

La Data SIS se genera del sistema del SIS, SIGEPS, a través de reportes mensuales generados en formato EXCEL, a nivel de establecimiento de salud (IPRESS).

Para facilitar una obtención de datos de forma oportuna, se implemento un script que permite realizar web scrapping al sistema, permitiendo la generación de los reportes mensuales correspondientes a las 147 IPRESS de la jurisdicción de la UE (RSAC), de forma automatizada, consolidando toda la data a nivel de UE.

```
from selenium import webdriver
import chromedriver_binary
import requests
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support.ui import Select
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.chrome.options import Options
from datetime import datetime

import ctypes
import time
import os
import pandas as pd
import sys
import subprocess
import win32com.client as win32
from datetime import datetime
import shutil
import openpyxl
import pytz
import glob

global IPRESS
IPRESS = [1298, 1316, 1317, 1318, 1291, 1300, 1302, 1303, 1301, 1304, 1296,
1313, 1315, 1310, 1235, 1246, 1245, 1420, 1433, 1434, 1441, 1432, 1421,
1431, 1436, 1437, 6891, 1435, 1438, 1422, 1440, 7407, 1418, 1442, 1236,
1248, 27843, 1261, 11677, 1287, 1286, 1284, 1289, 1285, 1288, 1262, 1299,
1319, 1320, 1321, 1419, 1425, 1423, 1424, 1428, 1429, 1426, 1427, 1430,
1297, 1312, 1314, 1311, 7744, 25066, 1253, 1255, 1242, 1256, 1241, 1254,
1239, 1243, 1292, 1293, 1307, 1306, 1305, 32056, 1325, 1326, 1339, 1340,
1336, 1378, 1342, 1341, 1337, 23969, 1234, 6890, 1244, 27464, 1295, 7722,
1309, 1259, 1275, 1279, 1273, 1276, 1274, 1277, 31488, 1322, 1330, 1328,
1327, 1329, 1294, 1308, 1238, 1249, 1250, 1252, 1237, 1247, 1251, 11023,
1323, 1333, 1331, 1332, 1343, 1258, 1271, 1268, 1269, 1270, 23338, 1272,
30346, 1260, 1280, 1281, 1290, 1257, 1264, 1263, 1267, 23488, 1324, 1335,
1334, 1233, 1282, 1283]

AÑO_SIGEPS = {2024:"15", 2023:"16", 2022:"17", 2021:"18", 2020:"19",
2019:"20", 2018:"21", 2017:"22", 2016:"23", 2015:"24", 2014:"25",
2013:"26", 2012:"27", 2011:"28", 2010:"29", 2009:"30", 2008:"31"}

def fecha_actual(ruta):

    libro = openpyxl.Workbook()
```

```

hoja = libro.active

formato_fecha = "DD/MM/YYYY"
formato_hora = "HH:MM"

now = datetime.now()
fecha = now.date()
hora = now.time()

hoja["A1"]="FECHA"
hoja["A2"].number_format = formato_fecha
hoja["A2"].value = fecha
hoja["B1"]="HORA"
hoja["B2"].number_format = formato_hora
hoja["B2"].value = hora

nombre_archivo = ruta + "\\_OUTPUT\\Actualización.xlsx"

if os.path.exists(nombre_archivo):
    os.remove(nombre_archivo)

if not os.path.exists(ruta + "\\_OUTPUT\\"):
    shutil.rmtree(ruta + "\\_OUTPUT\\")
    os.makedirs(ruta + "\\_OUTPUT\\")

libro.save(nombre_archivo)

def num_mes(nom_mes):

    if (nom_mes.upper()=="ENERO"):
        n = "01"
    elif (nom_mes.upper()=="FEBRERO"):
        n = "02"
    elif (nom_mes.upper()=="MARZO"):
        n = "03"
    elif (nom_mes.upper()=="ABRIL"):
        n = "04"
    elif (nom_mes.upper()=="MAYO"):
        n = "05"
    elif (nom_mes.upper()=="JUNIO"):
        n = "06"
    elif (nom_mes.upper()=="JULIO"):
        n = "07"
    elif (nom_mes.upper()=="AGOSTO"):
        n = "08"
    elif (nom_mes.upper()=="SETIEMBRE"):
        n = "09"
    elif (nom_mes.upper()=="OCTUBRE"):
        n = "10"
    elif (nom_mes.upper()=="NOVIEMBRE"):
        n = "11"
    elif (nom_mes.upper()=="DICIEMBRE"):
        n = "12"
    else:
        n = "00"

    return n

```

```

def mensaje():
    print("-----")
    print("DIGITACION FUA GENERATOR v1.5")
    print("-----")
    print("OFICINA DE SEGUROS")
    print("UE 1222 - RED DE SALUD AREQUIPA-CAYLLOMA")
    print("-----")
    print("© Vladimir Barrios Bejarano")
    print("Arequipa 2024\n")

def mensaje_error(msg_error):
    os.system("cls")
    mensaje()
    input((msg_error + " no válido, vuelva a intentarlo..."))
    engine.say(msg_error + " no válido, vuelva a intentarlo...")
    engine.runAndWait()
    os.system("cls")
    mensaje()
    print("Siga los siguientes pasos:\n")

def espera(segundos):
    inicio = time.time()
    while True:
        actual = time.time()
        if actual - inicio >= segundos:
            break

def selecciona_web(cadena_xpath):

    text_select = driver.find_element(By.XPATH,cadena_xpath)
    return(text_select.text)

def pausa_clic(cadena_xpath):

    wait = WebDriverWait(driver,3)

    while True:
        try:

            element =
wait.until(EC.element_to_be_clickable((By.XPATH,cadena_xpath)))

            if element.is_enabled():
                element.click()
                break
            else:
                time.sleep(2)

        except:
            time.sleep(2)

def cargar_Atenciones_IPRESS(IPRESS, v_Año, v_Mes):

    pausa_clic("//*[@id='divMenu']/div[3]/div[1]/div/div[2]/h4/a")

```

```

pausa_clic("//*[@id='collapse8']/div/a")

pausa_clic("//*[@id='mat-select-0']/div/div[2]/div")
pausa_clic("//*[@id='mat-option-0']")

for j in range(1):

    pausa_clic("//*[@id='mat-select-1']/div/div[2]/div")

    if j==0:
        pausa_clic("//*[@id='mat-option-189']/span")
    if j==1:
        pausa_clic("//*[@id='mat-option-193']/span")

    pausa_clic("//*[@id='mat-select-2']/div/div[2]/div")
    pausa_clic("//*[@id='mat-option-'+AÑO_SIGEPS.get(v_Año)+'"]")

    pausa_clic("//*[@id='mat-select-3']/div/div[2]/div")
    pausa_clic("//*[@id='mat-option-{str(v_Mes+2)}']")

for i in range(len(IPRESS)):

    if os.path.exists(path_prod):
        try:
            shutil.rmtree(path_prod)
            os.makedirs(path_prod)
        except OSError as e:
            print(f"Error: {e.filename} - {e.strerror}.")

    os.system("cls")
    mensaje()
    print("DESCARGANDO REPORTES DEL SISTEMA SIGEPS...")
    print("(" + str(i+1) + " de " + str(len(IPRESS)) + ")")

    elemento1 = driver.find_element(By.XPATH, "//*[@id='mat-input-2']")
    elemento1.send_keys(str(IPRESS[i]))

    pausa_clic("//*[@id='mat-input-2']")

    pausa_clic("//*[@id='mat-autocomplete-2']")

    pausa_clic("//*[@id='content-wrapper']/div/app-
default/div/div/form/div[3]/div[2]/button")

    pausa_clic("//*[@id='content-wrapper']/div/app-
default/div/div/form/div[2]/div/div/div[2]/div/div[6]/mat-form-
field/div/div[1]/div/i[1]")

    archivo = ""

    while len(archivo) == 0 and i < 147:

        archivo = glob.glob(f"{path_prod}/*.xlsx")

        for file in archivo:
            try:

```

```

        df = pd.read_excel(file, dtype=str)
    except ValueError as e:
        pass

    if j==0:
        df.to_excel(path_atenc+"\\"+str(IPRESS[i])+".xlsx",index=False)
    if j==1:
        df = df.assign(AÑO=v_Año, MES=v_Mes, IPRESS=IPRESS[i])
        df.to_excel(path_asist+"\\"+str(IPRESS[i])+".xlsx",index=False)

def presiona_mayusc():
    ctypes.windll.user32.keybd_event(0x14, 0, 0, 0) # Presiona la tecla
    ctypes.windll.user32.keybd_event(0x14, 0, 2, 0) # Suelta la tecla

def verifica_mayusc():
    key_state = ctypes.windll.user32.GetKeyState(0x14)
    return key_state & 0x0001 != 0

try:

    request = requests.get("http://sigeps.sis.gob.pe/",timeout=10)

except (requests.ConnectionError,requests.Timeout):

    print("No se pudo realizar la conexión con el aplicativo web SIGEPS")

else:

    path_temp = os.environ['TEMP']
    path_gen_py = os.path.join(path_temp, 'gen_py')

    if os.path.exists(path_gen_py):
        shutil.rmtree(path_gen_py)

    driver = webdriver.Chrome()
    path = "C:\\\\Digitacion_FUA_Generator"
    path_atenc = "C:\\\\Digitacion_FUA_Generator\\Atenciones EESS"
    path_asist = "C:\\\\Digitacion_FUA_Generator\\Asistenciales EESS"
    path_prod = "C:\\\\Digitacion_FUA_Generator\\Produccion"

    options = webdriver.ChromeOptions()
    prefs = {"download.default_directory": path_prod}
    options.add_argument("--unsafely-treat-insecure-origin-as-secure=http://sigeps.sis.gob.pe/")
    options.add_experimental_option("prefs", prefs)

    driver = webdriver.Chrome(options=options)
    driver.minimize_window()

    if os.path.exists(path_prod):
        shutil.rmtree(path_prod)
    if os.path.exists(path_atenc):
        shutil.rmtree(path_atenc)
    if os.path.exists(path_asist):
        shutil.rmtree(path_asist)

    os.makedirs(path_prod)

```

```

os.makedirs(path_atenc)
os.makedirs(path_asist)

mensaje()

engine = pyttsx3.init()

print("Siga los siguientes pasos:\n")
engine.say("Siga los siguientes pasos")
engine.runAndWait()

print("Ingrese las credenciales del SIGEPS en el navegador 'Chrome' y haga
clic en el botón 'Ingresar'\n")
engine.say("Ingrese las credenciales del SIGEPS en el navegador 'Chrome'
y haga clic en el boton Ingresar")
engine.runAndWait()

if verifica_mayusc():
    presiona_mayusc()

driver.maximize_window()
driver.get("http://sigeps.sis.gob.pe/")

espera(3)

w = win32.Dispatch('WScript.Shell')
#w.AppActivate('Chrome')
w.SendKeys('^I')
w.SendKeys('{BACKSPACE 100}')
if verifica_mayusc():

w.SendKeys("HTTP://SIGEPS.SIS.GOB.PE/sISerp/sISrEPORTESsIGEPS/RPTpRE
LIMINAR")
else:

w.SendKeys("http://sigeps.sis.gob.pe/SisERP/SisReportesSigeps/rptPrelimin
ar")
w.SendKeys('{ENTER}')

pausa_clic("//*[@id='username']")

driver.get("http://sigeps.sis.gob.pe/")

os.system("cls")
mensaje()
engine.say("Cuando se encuentre en el módulo principal del sistema
SIGEPS, presione ÉNTER en la consola de DIGITACION FUA GENERATOR para
continuar")
engine.runAndWait()
input("Cuando se encuentre en el módulo principal del sistema SIGEPS,
presione ENTER en la consola DE DIGITACION FUA GENERATOR para
continuar...")

os.system("cls")
mensaje()
print("Siga los siguientes pasos:\n")

```

```
engine.say("Ingrese el año a consultar (dentro del rango del año 2008 al 2024): ")
engine.runAndWait()
```

```
while True:
```

```
    consulta_año = int(input("Ingrese el año a consultar (2008-2024): "))
```

```
    if consulta_año >= 2008 and consulta_año <= 2024:
```

```
        break
```

```
    else:
```

```
        mensaje_error("Año")
```

```
engine.say("Ingrese el mes a consultar (del rango del 1 al 12):")
```

```
engine.runAndWait()
```

```
os.system("cls")
```

```
mensaje()
```

```
print("Siga los siguientes pasos:\n")
```

```
while True:
```

```
    consulta_mes = int(input("Ingrese el mes a consultar (1-12): "))
```

```
    if ((consulta_mes >= 1 and consulta_mes <= 12 and consulta_año <
datetime.now().year) or (consulta_mes >= 1 and consulta_mes <=
datetime.now().month and consulta_año == datetime.now().year)):
```

```
        break
```

```
    else:
```

```
        mensaje_error("Mes")
```

```
os.system("cls")
```

```
mensaje()
```

```
fecha_actual(path)
```

```
engine.say("Presione ÉNTER para iniciar con la descarga de la data...")
```

```
engine.runAndWait()
```

```
input("Presione ENTER para iniciar con la descarga de la data...")
```

```
cargar_Atenciones_IPRESS(IPRESS, consulta_año, consulta_mes)
```

```
os.system("cls")
```

```
mensaje()
```

```
print("Carga finalizada exitosamente...")
```

```
time.sleep(2)
```

```
os.system("cls")
```

```
mensaje()
```

```
subprocess.call("taskkill /F /IM EXCEL.exe",shell=True)
```

```
os.system("cls")
```

```
excel = win32.gencache.EnsureDispatch('Excel.Application')
```

```

os.system("cls")

def csv_data(ruta:str,extension:str):

    files = glob.glob(ruta + "\\*" + extension, recursive=True)
    for f in files:
        os.system("cls")
        print(" ► " + f)
        df = pd.read_csv(f, sep=';', encoding='mbcs')
        #df = df.drop(['AFILIADO', 'HISTORIA'], axis=1)
        df.to_excel(f[:-3] + '.xlsx', index=False)

files = []
csv_data(path_atenc, '.csv')
csv_data(path_asist, '.csv')

def excel_data(ruta:str,extension:str):
    files = glob.glob(ruta + "\\*" + extension, recursive=True)
    return files

excel.Application.Quit()
files = []

for j in range(2):

    if j==0:
        files=excel_data(path_atenc, '.xls')
    if j==1:
        files=excel_data(path_asist, '.xls')

    for f in files:

        if f.find(path_atenc)>-1 or f.find(path_asist)>-1:
            if os.path_atenc.exists(f + ".x") or os.path_asist.exists(f + ".x"):
                os.remove(f + ".x")
            wb = excel.Workbooks.Open(f)
            wb.SaveAs(f + ".x", 51)
            wb.Close()
            os.remove(f)
            os.system("cls")
            print("ACTUALIZANDO DATA...")
            os.system("cls")
            print(" ► " + f)

excel.Application.Quit()
files = []

for j in range(2):

    if j==0:
        files=excel_data(path_atenc, '.xlsx')
    if j==1:
        files=excel_data(path_asist, '.xlsx')

    os.system("cls")

    df = pd.DataFrame()

```



```

all_data = pd.DataFrame()

try:

    for f in files:
        print(f)
        if f.find(path_atenc)>-1 or f.find(path_asist)>-1:
            mensaje()
            print("LECTURA DE DATA ACTUALIZADA (Espere por favor)...")
            print('► ' + f)
            df = pd.read_excel(f,dtype=str)
            #df = df.drop(['AFILIADO','HISTORIA'],axis=1)
            all_data = pd.concat([all_data,df],axis=0)
            os.system("cls")

    if not os.path.exists(path + "\\_OUTPUT\\"):
        os.makedirs(path + "\\_OUTPUT\\")

    mensaje()
    print("GENERANDO TABLA...")

    os.system("cls")
    mensaje()

    año=selecciona_web("//*[@id='mat-select-2']")
    mes=num_mes(selecciona_web("//*[@id='mat-select-3']"))

    if j==0:
        all_data.to_excel(path +
        '\\_OUTPUT\\_' + año + '_' + mes + '_PROFESIONALES_EESS.xlsx',
        sheet_name='Sheet1',index=False)
        if j==1:
            all_data.to_excel(path +
            '\\_OUTPUT\\_' + año + '_' + mes + '_Asistenciales_EESS.xlsx',
            sheet_name='Sheet1',index=False)

        driver.close()
        driver.quit()

        engine.say("¡ PROCESO TERMINADO EXITÓSAMENTE !")
        engine.runAndWait()

        os.system("cls")
        mensaje()
        print("¡ PROCESO TERMINADO EXITÓSAMENTE !\n")
        print(r"El archivo procesado se guardó en
C:\Digitacion_FUA_Generator\_OUTPUT")

        engine.say("\nEl archivo procesado se guardó en la partición C,
carpeta Digitacion FUA Generator, dentro de la carpeta subguión OUTPUT")
        engine.runAndWait()
        time.sleep(2)

except Exception as E:

    os.system("cls")
    mensaje()

```

```
print("¡ ERROR EN LA GENERACIÓN DE TABLAS !")
print(E)
print("")

engine.say("Presione ÉNTER para salir. Oficina de Seguros de la Red de
Salud Arequipa Caylloma")
engine.runAndWait()
input("\n[Presione ENTER para salir...]")
sys.exit()
```

Adicionalmente, se vió la necesidad de hacer uso del dataset CIE-10, el cual contiene el listado de códigos de diagnósticos de la Clasificación Internacional de Enfermedades, Decima Edición, la cual se obtuvo de datos públicos proporcionados por el Ministerio de Salud.

Además, se incluyó un dataset IPRESS, el cual contiene el listado de los 147 establecimientos de salud (IPRESS) del ámbito de la jurisdicción de la Red de Salud Arequipa Caylloma, con sus respectivos códigos de IPRESS que se maneja a nivel de HIS y SIS, lo cual permitirá identificar los establecimientos en posteriores análisis.

2. ALMACENAMIENTO

La data obtenida se almacena en un repositorio en la nube (Google Drive) para que se encuentre disponible de forma oportuna para un posterior procesamiento de datos.

🔍 Buscar en Drive













Mi unidad > Recuperacion Informa... > Data ▾

Tipo ▾

Personas ▾

Modificado ▾

🔊 ¡Nuevo! Combinaciones de teclas Las combinaciones de teclas de Drive se han actualizado

| Nombre ↑ | Propietario |
|--|--|
|  _ANÁLISIS |  yo |
|  _PRE-PROCESAMIENTO |  yo |
|  CIE-10 |  yo |
|  HIS |  yo |
|  IPRESS |  yo |
|  SIS |  yo |

... > Data > SIS ▾

Tipo ▾

Personas ▾

Modificado ▾

🔊 ¡Nuevo! Combinaciones de teclas Las combinaciones de teclas de Drive se han actualizado

| Nombre ↑ | Propietario |
|---|--|
|  2024_01_PROFESIONALES_EESS.xlsx |  yo |
|  2024_02_PROFESIONALES_EESS.xlsx |  yo |
|  2024_03_PROFESIONALES_EESS.xlsx |  yo |
|  2024_04_PROFESIONALES_EESS.xlsx |  yo |

... > Data > HIS ▾

Tipo ▾

Personas ▾

Modificado ▾

🔔 ¡Nuevo! Combinaciones de teclas Las combinaciones de teclas de Drive se han actualizado

| Nombre | Propietario |
|---|--|
|  01.ENERO.xlsx |  yo |
|  02.FEBRERO.xlsx |  yo |
|  03.MARZO.xlsx |  yo |
|  04.ABRIL.xlsx |  yo |


... > Data > IPRESS ▾

Tipo ▾

Personas ▾

Modificado ▾

🔔 ¡Nuevo! Combinaciones de teclas Las combinaciones de teclas de Drive se han actualizado

| Nombre | Propietario |
|---|--|
|  COD_IPRESS.xlsx |  yo |


... > Data > CIE-10 ▾

Tipo ▾

Personas ▾

Modificado ▾

🔔 ¡Nuevo! Combinaciones de teclas Las combinaciones de teclas de Drive se han actualizado

| Nombre | Propietario |
|--|--|
|  CIE-10_con_Gravedad.xlsx |  yo |

3. PROCESAMIENTO

- Consolidando la Data SIS cargando dimensiones de interés.

```
import os

import pandas as pd

folder_path_sis = '/content/drive/MyDrive/Recuperacion
Informacion_Examen Final/Data/SIS'

dfs_sis = []

for file_name in os.listdir(folder_path_sis):
    if file_name.endswith('.xlsx'):
        file_path = os.path.join(folder_path_sis, file_name)

        df = pd.read_excel(file_path, usecols=[
            'F. ATENCION', 'DOCUMENTO', 'EDAD', 'SEXO',
            'DNI PROFESIONAL', 'EESS CODIGO', 'SERVICIO', 'TIPO PROFESIONAL'
        ])

        dfs_sis.append(df)

consolidated_sis_df = pd.concat(dfs_sis, ignore_index=True)

consolidated_sis_df.head()

consolidated_sis_df.to_csv('/content/drive/MyDrive/Recuperacion
Informacion_Examen Final/Data/_PRE-
PROCESAMIENTO/DATA_SIS_Consolidada.csv', index=False)
```

Consolidando Data HIS con dimensiones de interés

```
import os
```

```
import pandas as pd
```

```
folder_path_his = '/content/drive/MyDrive/Recuperacion  
Informacion_Examen Final/Data/HIS'
```

```
dfs_his = []
```

```
for file_name in os.listdir(folder_path_his):
```

```
    if file_name.endswith('.xlsx'):
```

```
        file_path = os.path.join(folder_path_his, file_name)
```

```
        df_his = pd.read_excel(file_path, usecols=[  
            'fecha atención', 'dni paciente', 'dni personal',  
            'ipress',  
            'codigo1', 'codigo2', 'codigo3', 'codigo4', 'codigo5',  
            'codigo6', 'codigo7', 'codigo8', 'codigo9', 'codigo10',  
            'codigo11', 'codigo12', 'codigo13', 'codigo14', 'codigo15',  
            'codigo16', 'codigo17', 'codigo18', 'codigo19', 'codigo20',  
            'codigo21', 'codigo22', 'codigo23', 'codigo24', 'codigo25',  
            'codigo26', 'codigo27', 'codigo28'  
        ])
```

```
        dfs_his.append(df_his)
```

```
consolidated_his_df = pd.concat(dfs_his, ignore_index=True)
```

```
consolidated_his_df.head()
```

```
consolidated_his_df.to_csv('/content/drive/MyDrive/Recuperacion
Informacion_Examen Final/Data/_PRE-
PROCESAMIENTO/DATA_HIS_Consolidada.csv', index=False)
```

- Limpieza de Data HIS consolidada, estandarizando los códigos de 1 al 28, en código, para convertir la data semiestructurada en data estructura para facilitar su procesamiento.

```
import os
```

```
import pandas as pd
```

```
cie10_path = '/content/drive/MyDrive/Recuperacion Informacion_Examen
Final/Data/CIE-10/CIE-10_con_Gravedad.xlsx'
```

```
data_his_path = '/content/drive/MyDrive/Recuperacion
Informacion_Examen Final/Data/_PRE-
PROCESAMIENTO/DATA_HIS_Consolidada.csv'
```

```
# Cargar los datos del CIE-10
```

```
cie10_df = pd.read_excel(cie10_path, usecols=['IdCiex', 'Ciex', 'Gravedad'])
```

```
# Cargar los datos del HIS consolidado
```

```
data_his_df = pd.read_csv(data_his_path)
```

```
codes_df = data_his_df.melt(
    id_vars=['fecha atención', 'dni paciente', 'dni personal', 'ipress'],
    value_vars=[f'codigo{i}' for i in range(1, 29)],
    var_name='codigo_columna',
    value_name='codigo'
)
```

```
merged_df = codes_df.merge(cie10_df, left_on='codigo', right_on='IdCiex',
how='left')
```

```
filtered_df = merged_df.dropna(subset=['Gravedad'])
```

```
idx = filtered_df.groupby(['fecha atención', 'dni paciente', 'dni personal',  
'ipress'])['Gravedad'].idxmax()
```

```
final_df = filtered_df.loc[idx]
```

```
final_df['Diagnostico'] = final_df['codigo']
```

```
final_his_df = final_df[['fecha atención', 'dni paciente', 'dni personal',  
'ipress', 'Diagnostico', 'Ciex']]
```

```
final_his_df.to_csv('/content/drive/MyDrive/Recuperacion  
Informacion_Examen Final/Data/_PRE-  
PROCESAMIENTO/DATA_HIS_Consolidada_Preproceso_1.csv', index=False)
```

- Limpieza de Data SIS consolidada para establecer en base a los patrones que permitan agregar datos relevantes para el análisis (Diagnóstico, CieX)

```
import pandas as pd
```

```
# Cargar los datos
```

```
sis_data_path = '/content/drive/MyDrive/Recuperacion  
Informacion_Examen Final/Data/_PRE-  
PROCESAMIENTO/DATA_SIS_Consolidada.csv'
```

```
his_data_path = '/content/drive/MyDrive/Recuperacion  
Informacion_Examen Final/Data/_PRE-  
PROCESAMIENTO/DATA_HIS_Consolidada_Preproceso_1.csv'
```

```
ipress_data_path = '/content/drive/MyDrive/Recuperacion  
Informacion_Examen Final/Data/IPRESS/COD_IPRESS.xlsx'
```

```
# Leer datos de IPRESS
```

```
ipress_df = pd.read_excel(ipress_data_path)
```

```
# Leer datos de SIS y convertir a string
```



```

sis_df = pd.read_csv(sis_data_path)

sis_df = sis_df.astype(str)

ipress_df = ipress_df.astype(str)

# Merge para obtener COD_IPRESS en SIS

sis_df = pd.merge(sis_df, ipress_df, left_on='EESS CODIGO', right_on='EESS',
how='left')

sis_df.rename(columns={'COD_IPRESS': 'IPRESS SIS'}, inplace=True)

# Limpieza y transformación de los datos en SIS

sis_df['F. ATENCION'] = sis_df['F. ATENCION'].str[:10]

sis_df['DOCUMENTO'] =
sis_df['DOCUMENTO'].str.split('.').str[0].str.zfill(8).str.strip()

sis_df['DNI PROFESIONAL'] = sis_df['DNI
PROFESIONAL'].str.split('.').str[0].str.zfill(8).str.strip()

sis_df['IPRESS SIS'] = sis_df['IPRESS SIS'].str.strip()

# Generar la variable 'sis'

sis_df['sis'] = sis_df['F. ATENCION'] + "." + sis_df['DOCUMENTO'] + "." +
sis_df['IPRESS SIS'] + "." + sis_df['DNI PROFESIONAL']

# Leer y procesar datos de HIS

his_df = pd.read_csv(his_data_path)

his_df = his_df.astype(str)

his_df['fecha atención'] = his_df['fecha atención'].str.strip()

his_df['dni paciente'] = his_df['dni
paciente'].str.split('.').str[0].str.zfill(8).str.strip()

his_df['ipress'] = his_df['ipress'].str.strip()

his_df['dni personal'] = his_df['dni
personal'].str.split('.').str[0].str.zfill(8).str.strip()

# Generar la variable 'his'

his_df['his'] = his_df['fecha atención'] + "." + his_df['dni paciente'] + "." +
his_df['ipress'] + "." + his_df['dni personal']

```

```
# Asegurarse de que las variables 'sis' y 'his' estén bien formateadas

sis_df['sis'] = sis_df['sis'].str.strip()
his_df['his'] = his_df['his'].str.strip()


merged_df = pd.merge(sis_df, his_df[['his', 'Diagnostico', 'Ciex']],
left_on='sis', right_on='his', how='left')


merged_df = merged_df[merged_df['his'].notna()]
merged_df.drop(columns=['his'], inplace=True)


merged_df.to_csv('/content/drive/MyDrive/Recuperacion
Informacion_Examen Final/Data/_PRE-
PROCESAMIENTO/DATA_SIS_Consolidada_Preproceso_2.csv', index=False)


print("Merge completado y datos guardados.")
```

- Data procesada

... > Data > _PRE-PROCESAMIENTO ▾

Tipo ▾

Personas ▾

Modificado ▾

🔊 ¡Nuevo! Combinaciones de teclas Las combinaciones de teclas de Drive se han actualizado

Nombre ↑

Propietario



DATA_HIS_Consolidada_Preproceso_1.csv



yo



DATA_HIS_Consolidada.csv



yo



DATA_SIS_Consolidada_Preproceso_2.csv



yo



DATA_SIS_Consolidada.csv



yo

X. REFERENCIAS

Marcos Valdez, A. J., Navarro Ortiz, E. G., Quinteros Peralta, R. E., Tirado Julca, J. J., Valentin Ricaldi, D. F., & Calderon-Vilca, H. D. (2023). Aprendizaje automático para predicción de anemia en niños menores de 5 años mediante el análisis de su estado de nutrición usando minería de datos. *Computación y Sistemas*, 27(3), 749-768.